Int. J. Advance Soft Compu. Appl, Vol. 5, No. 3, December 2013
ISSN 2074-8523; Copyright © SCRG Publication, 2013

Multi-Criteria Architecture Style Selection for Precision Farming Software Product Lines Using Fuzzy AHP

Mohd Z.M. Zaki [1], Dayang N.A. Jawawi [1], Norazian M. Hamdan [2], Shahliza Abd. Halim [1], Rosbi Mamat [3], Fairuz S. Mahat [1], and Nur Athirah Omar [1]

[1] Software Engineering Department, Faculty of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia
{zulkiflizaki, dayang, shahliza}@utm.my, {fsafwan,nurathirahomar}@gmail.com
[2] Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia
mhnorazian@fit.unimas.my
[3] Department of Control Engineering and Mechatronic Engineering, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Johor, Malaysia
rosbi@fke.utm.my

Abstract

Precision Farming (PF) system is an alternative and innovative approach to improve the quality and production of crop yields. However, due to heterogeneity and user demands, PF system complexity has become higher. As such, software complexity has always been an issue in software development, especially for larger systems with innovative functionalities. One solution by which to reduce the problem of software complexity is by incorporating software reuse. Software Product Line (SPL) is a strategic reuse approach, which targets common artefacts for its product line while having a variability management mechanism to cater for variability in individual applications. This research proposes an integrated approach of SPL with architecture style selection and component-based design for the precision farming domain. The focus of this paper is to highlight the process of architecture style selection in the proposed approach, which involves a multi-criteria design decision. The selection process uses a fuzzy analytic hierarchy process (fuzzy AHP) in order to select the best architectural style, which can fulfil most of the sought-after criteria for precision farming product line application.

Keywords: *Precision Farming, Software Product Lines, Software Architecture, Fuzzy AHP.*

1 Introduction

Towards the 21st century, information technologies have been rapidly advancing and have since been applied to many fields, including agriculture. In the past, the agricultural-related activities were performed in a traditional way mostly involving human labour to operate diverse equipment and machines. Farmers need to constantly visit the crop fields to monitor the conditions and the data were measured manually, crop by crop. Decisions on harvesting, as well as the suitable amount of fertilizers and pesticides, can only be made after gathering enough information from the crop field. These traditional ways were ineffective and time-consuming, especially if they involved monitoring large-sized crop fields. Fortunately, with the advancement of information technologies, these agricultural activities were modernized by enabling automation to replace manual operations. Hardware devices like sensors and actuators are deployed on the crop field to gather and measure data, and the software system is used to process the data for decision-making. As a result, the need for human labour has been decreased and the time needed for data collection has been reduced. This new method is known as Precision Farming (PF).

However, with the existence of many devices, sensors and actuators, the system complexity has become higher. The issue of complexity has been quite common in the computing field for some decades. Among the contributing factors are customer demands for innovative system functionalities, many kinds of platforms created by vendors, and the ever-changing requirements by the customers. Therefore, the code size and error rates will increase and it will become difficult to maintain.

Software reusability can be an appropriate strategy by which to solve this complexity issue [1][2]. Software Product Line (SPL) is one of a number of emerging paradigms that promotes reusability of software assets like components, architectures, designs, data and modules [2]. SPL in software development aims to produce software at lower costs and in a shorter time without neglecting commonalities and variability in similar applications of the product lines [3]. Although there are many SPL methods available, such as COPA [9], FAST [10], FORM [11], KobrA [12] and QADA [13], each of these methods has their own advantages. However, most of them lack methods documentation and therefore there are no concrete descriptions of processes involved. The problem also applies to the design decision involved in the building of the PL architecture, which is the most important artefact in the SPL process.

In SPL, the most important concept is the identified attributes of a system, known as features [5] that serve as a representation of reusable components and requirements of a SPL by which to exploit the commonality and core assets and manage their variability. Although SPL methods already cover the identification and recognition of reusable core assets, there is still a gap between analysis and design whereby the identified core assets are not formalized into a proper design model. The lack of a systematic approach by which to compose and integrate the

identified reusable components has made the development of a functional PF system undesirable. This is because most of the PF systems are manually developed from scratch. Therefore, the introduction of suitable software architecture could help to map PF software requirements with regard to architectural design and ensure that both functional and non-functional requirements are met.

Nevertheless, each of the PF systems has different requirements. Thus, selection of a suitable software architecture style is an important process in PF software development because this choice could affect the quality of the final product [15]. The objective of architecture style selection is to identify an architecture style with the highest potential for meeting PF system requirements.

The aim of this paper is to develop a PF system using enhanced SPL methods with software architecture style from the selection using a multi-criteria selection method. This will start from the analysis phase and extend until the initial architecture design in the design phase. The organization of this paper is as the following sections. Section 1 introduces the PF system, SPL methods, multi-criteria selection method and software architecture style. Section 2 describes the methodology, which is proposed for the paper. Section 3 explains the analysis phase using SPL methods. Section 4 elaborates on the selection phase using a multi-criteria selection method and Section 5 describes the initial design of the PF system using the selected architecture style. Finally, Section 6 concludes the objectives of the paper and reports the results found in the research.

2 Integrated Approach

This paper has proposed an integration approach for PF system development involving the integrated SPL framework, multi-criteria selection method and architecture design. The methodology is proposed and illustrated in Fig. 1. The methodology takes into consideration several phases, namely: analysis phase, selection phase and design phase respectively.

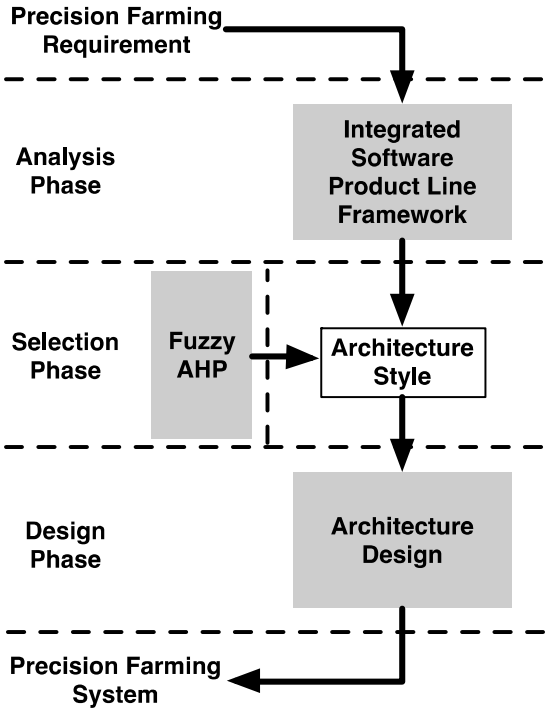

Fig. 1: Integrated Approach

In the analysis phase, the requirements of the PF system are analysed using an integrated SPL framework. This framework consists of a combination of several methods by which to produce a feature model. Based on the feature model, reference architecture is produced. This reference architecture is important in order to identify core assets that are reusable for the PF system. A software architecture style is selected in the selection phase. The selection process is performed using the fuzzy AHP method. Multiple criteria are extracted from PF system requirements to enable the selection of suitable architecture style. The design phase handles the design of the reusable assets in the form of software pattern, based on the selection phase earlier. As a result, the PF system has been designed.

Basically, this integration is important in order to improve the current SPL method. The current SPL method does not provide a concrete description on the process by which to map architecture core assets to a concrete detailed design of the reusable components. The mapping is important to enable the model-to-code transformation. This integration can enable a systematic composition and integration process for developing the PF system from identified reusable components and not from scratch.

3 Analysis Phase

The most widely-used method for analysing a domain using features is Feature-Oriented Domain Analysis (FODA) [5]. The main motive for using FODA is to implement domain analysis in a systematic manner [6]. In FODA, those features are constructed into a feature tree or feature model. The feature model is composed of a hierarchy of features, with each branch holding mandatory, optional, or mutually exclusive conditions. The FODA method will be used to document requirement artefacts. The output will be a feature model of a PF system based on SPL. Therefore, the FODA method is useful to the Analysis Phase. In order to continue with the Design Phase and Implementation Phase, the FODA method is extended to the Feature-Oriented Reuse Method (FORM) [7]. However, the obvious weakness of FORM is that it does not provide concrete description on the process to map features to architecture styles. Further, it does not focus on providing a clear transition between feature model and architecture, instead only concretizing the FODA processes of analysis and design from a marketing perspective.

In order to map the requirements into architectural components, the Feature-Architecture Mapping (FArM) method is used [4][8]. To address the feature variability of software component, the FArM method is integrated with Feature Dependency Analysis (FDA) method processes.

3.1 Overview of Integrated SPL Methods

The FDA method was introduced in order to specifically address the handling of feature-oriented commonality and variability in an SPL [11]. In feature-oriented SPL, the structural and configuration dependencies can be addressed using feature modelling. However, it is also crucial to address the operational dependencies in software development as these dependencies represent the relationship between features during the operation of the system. Therefore, FDA's main objective is to represent the operational dependencies in software development.

The method that will be used for mapping requirements into feature model is the FODA method. FODA is chosen because it focuses mainly on the analysis of the selected domain [5], which in this case is the PF application. The input for FODA is the requirements collected from customers, stakeholders, as well as developers. However, in this paper, the requirements will be extracted from six case studies of PF based on WSN technologies. The requirements will be transformed into features and will be organized in a structured diagram called the feature model.

3.2 Domain Analysis Using FODA

The feature model from a study done by [12] is investigated so as to study the feature model's standard framework, which involves four layers, specifically: capability, operating environment, domain technology, and implementation

technique layers.

Based on the selected domain, three case studies are selected for on-farm PF and another three case studies for greenhouses PF. The case studies are selected from past research papers of PF application, using wireless technologies. The topic of discussion in these case studies includes: the systems for managing and monitoring PF activities, the hardware components that are used to achieve precise farming, the functionality requirements on each on-farm and greenhouse PF application. Each case study is investigated in order to extract the main components and requirements of PF based on its wireless environment. The requirements from on-farm PF are then compared to those from greenhouses PF in order to identify their respective similarities and differences. These similar and different requirements are then used for developing the feature model.

After all the features have been extracted from the selected case studies, they will then be compared in order to identify common features as well as variable features. They are then categorized in four layers, namely: Capability, Operating Environment, Domain Technology, and Implementation Technique layers. Other necessary requirements from on-farm PF that can be adopted are image capturing and location sensing. This result is presented in Table 1.

Based on the feature categories that have been produced in Table 1, a feature model is created which can be referred to in [13]. Relationships and variability features are represented to provide a clear understanding of communication between those domain features.

The transformations are done in two steps, namely: extracting the super-features and then organizing the super-features and sub-features. The first step of the transformation is to extract the super-features from all the layers in the feature model. The extracted super-features must be meaningful and represent the functional and non-functional features of the PF software PL. The next step is to organize the super-features and their sub-features. The super-features and their sub-features are organized into a hierarchical structure or tree diagram. Their relationships are shown using, specifically: Composed-Of, Generalization, Implemented-By, and Required notations. The features variability such as optional and alternative are also shown using notations.

Table 1: Feature Categories for PF SPL

Layer	Feature Group	Features
Capability Features	Service	Input Application • Fertilizers • Pesticides
		Field Environment Sensing • Soil • Climate • Water
		Location Sensing
		Harvesting Mechanism
		Image capturing
	Operation	Monitor and Control Operation
	Non-Functional Property	Usability
		Security
Operating Environment Features	Software/ Hardware Interface and Platform	Communication • Telephone • Internet • Bluetooth
		Detection Devices • Soil Sensor • Environment Sensor • Water Sensor
		Action Devices • Light • Fan • Sprinkler • Input Pump • Humidifier • Heater • CO_2 Pump
Domain Technology Features	Domain Specific Methods	Sensing Data • Discrete Value • Continuous Value
		Responding Strategy • Sequential • Priority
Implementation Technique Features	Design and Implementation Decisions	Connection • TCP • UDP
		Location • Manual • Automatic (GPS)

3.3 Transformation Using Integrated FArM and FDA

The next phase is to use the selected feature-architecture mapping method (FArM) to map the feature model onto architectural components. By using FArM, there will be two outputs, namely: final transformed feature model and reference architecture. In order to produce the reference architecture, the FDA method will also be applied to the FArM method so as to handle the feature variability issue. These two outputs will be used for creating the PF software architecture and are ready for software reuse. The main goal for designing the architectural components using FDA is to represent all three variable features, namely: alternative, OR, and optional.

The Transform Feature Model phase contains four transformations, namely: Non-Architecture-Related (NAR) and Quality Features, Architectural Requirements, Interacts Relations, and Hierarchy Relations as shown in Fig. 2. NAR features consist of features that do not have any direct impact on the software system. During NAR features transformation, the features are removed from the feature model. Quality feature are those non-functional requirements of the software system. The transformations use the transformation process of quality features into functional implementation of the quality features.

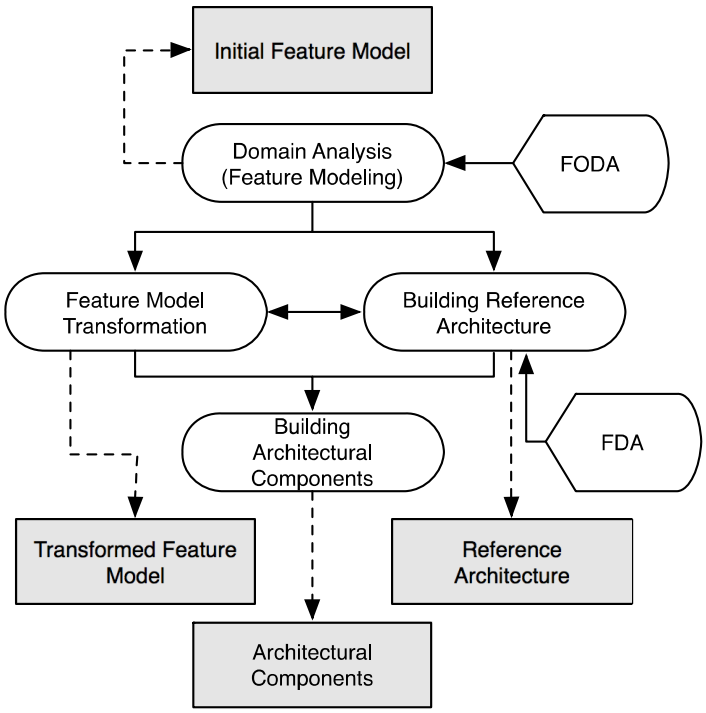

Fig. 2: Enhanced Integrated FArM Product Line Framework

The Architectural Requirements are the functional requirements of the software system. During this phase, the transformed feature model only contains the functional features. The term Interacts Relations refers to the communication and dependencies between features. The transformation of Interacts Relations may lead to the addition of new features or the integration of some features to other features. The term Hierarchy Relations refers to the relationships between super-features and their sub-features. The transformed feature model will display the features in a hierarchical manner. The four transformations are done in *n* number of iterations and parallel to the Building Reference Architecture phase. The developers may revisit previous transformations if necessary and then proceed through the rest of the transformations in the given order. Each transformation can lead to adding new features, integrating existing features to other features, dividing features, and reordering the hierarchy of the feature model respectively. During the Building Reference Architecture phase, the component specifications of the feature are derived. The PF Reference Architecture is illustrated in Fig. 3.

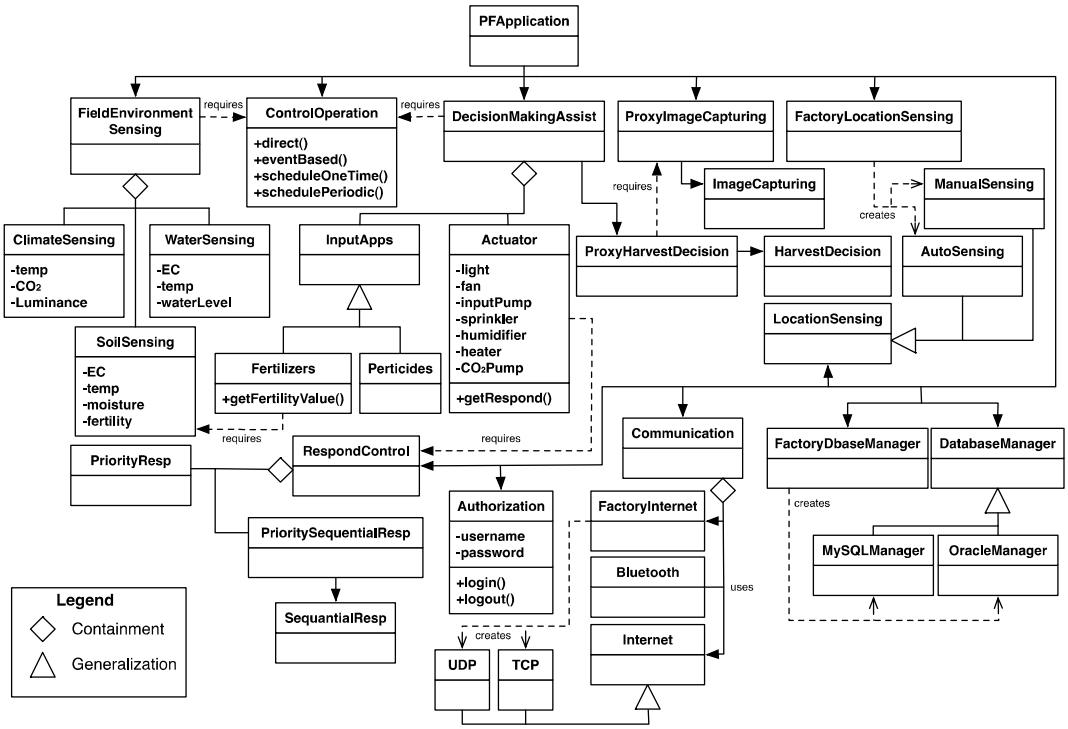

Fig. 3: PF Reference Architecture built using Enhanced FArM Method

4 Architecture Style Selection with Fuzzy AHP

Reference architecture is the software architecture that provides the common structures, components and their relationships to the existing systems in a particular domain [22]. Thus, the reference architecture generated from the Integrated Approach described in the previous sections is comprised of only the logical architecture based on the functional requirements identified in domain analysis. Focusing merely on the functional requirements is not enough where the overall quality of the software has to be considered. Quality attributes play an important role in software architecture where they affect the overall factors related to the software such as: run-time behaviour, system design, and user experience using the software [23]. Consequently, the appropriate architecture style is chosen as it already has the best practise in terms of knowledge in architectural development by the experts, and also the style already incorporates a certain degree of quality attributes.

Related works show that there are different architectural styles used in the domain of precision farming. An analysis has been done in the target domain to identify the architectural style used by researchers for precision agriculture software development. The analysis reveals three of the most prominent architecture style alternatives, specifically: layered architecture style [24][25], centralized architecture style [26] and client-server architecture style [27]. Due to the intuitive judgement involved in the process of selecting a suitable architecture style, and the challenges in fulfilling the variability criteria in SPL which have different functions and quality attributes concentration, different techniques are used to solve the problem. Analytical Hierarchy Process (AHP), a technique which facilitates the multi criteria decision-making, has been used by researchers in the selection of architecture styles [20][28] where AHP provides an overall ranking of architectural style based on the predetermined criteria given to it. There is also another research which concentrates on using the fuzzy model to help in the decision of architecture selection [21]. As in this paper, we concentrate on the hybridization of this technique where the fuzzy model handles the imprecise judgments made during architecture style selection, while the AHP will assist in the pair wise comparison of the architecture styles.

Prior to the use of Fuzzy AHP, the criteria and sub criteria for architecture style selection have to be determined as an input to the technique. The first criterion is reusability in product line application. Reusability involves two sub criteria, namely: common and variable components. All products in the product line use common components and variability components are used to cater for certain degrees of differences between similar products in a product line. Furthermore, architecture style selection is basically related with the non-functional criteria suitable for the application to be developed. Therefore, for the precision farming product line, we have identified two more criteria, namely: efficiency and flexibility. Efficiency has the sub criteria of time and resource utilization, which is based on ISO/IEC 9126 documentation. Both sub criteria are

important for the selection of architecture style, as the domain of precision farming requires different types of embedded hardware such as sensors for humidity and temperature to be deployed in the architecture, which will consequently affect the timing and memory utilization of the developed product. Flexibility has three sub criteria, specifically: change in algorithm, change in data representation and change in function [21]. These sub-criteria are suitable for product line application due to the variability aspect, which requires the architecture style ability to accommodate to changes either in its algorithm, its data representation or its function.

4.1 AHP criteria hierarchy

AHP is one of the most extensively used multi-criteria decision-making methods, and is a mathematical decision–making technique proposed by [16]. AHP can handle problems involving the evaluation of both tangible and intangible criteria and subsequently yield sensible numerical results. Among the researchers using AHP for architecture selection purpose are [20]. However, conventional AHP still cannot reflect the human thinking style, therefore AHP is extended using fuzzy logic to solve the fuzzy problems encountered in hierarchy [17][18].

In AHP, the hierarchy of criteria and sub-criteria described previously are mapped into a decision tree. Flexibility, performance and reusability are divided into several sub-criteria respectively. Sub-criteria for flexibility include: change in algorithm (CiA), change in function (CiF) and change in data representation (CiDR). Sub-criteria for performance consist of timing (Ti), resource utilization (RU) and efficiency (Ef). Sub-criteria for reusability comprise common (Co) and optional (Op). The alternative architecture styles are centralized, client-server and layered. The hierarchy is as shown in Fig. 4.

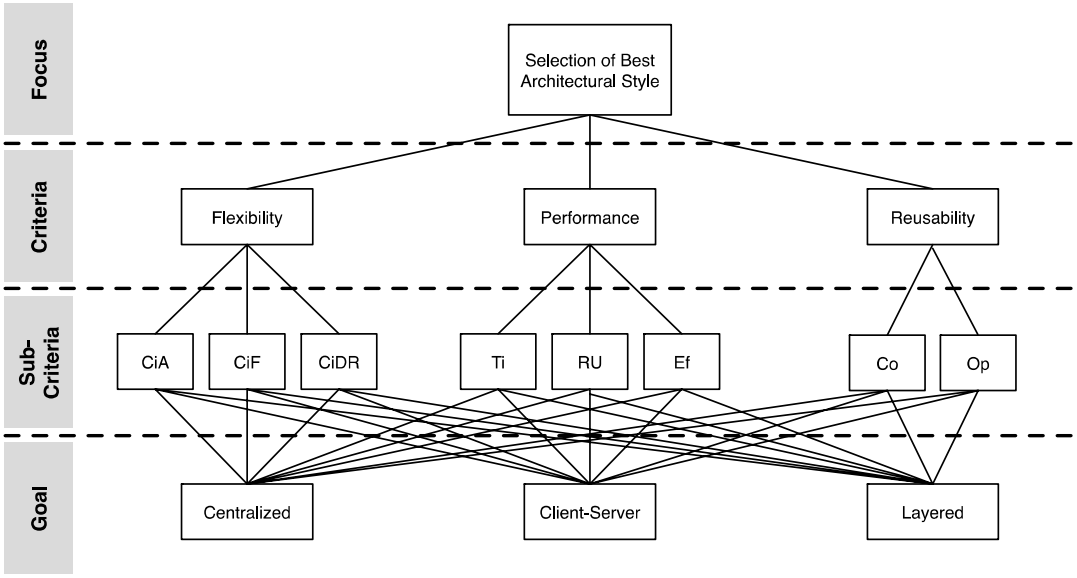

Fig. 4: The hierarchy of suitable software architecture style

4.2 Selection process using fuzzy AHP

The prioritization process of those criteria and sub-criteria is important in order to show the relationship of each element to decision-making. Later, a pair-wise comparison of each element is performed. The determination of this pair-wise comparison is required to be performed by various stakeholders. These comparisons are conducted based on the rules of AHP fundamental scale so as to measure the relative importance of each element [16][19].

However, in relation to fuzzy AHP, the fuzzy evaluation matrix uses a different scale based on a triangular fuzzy number. Based on this triangular fuzzy number, via pair-wise comparison of weighted matrix, a fuzzy evaluation matrix $A = (a_{ij})_{n \times m}$ is constructed. The pair-wise evaluation scale can be represented using the triangular fuzzy number. For the estimation of the importance of these criteria, the Fuzzy AHP is utilized. Let say that $\tilde{A}$ represents a fuzzified reciprocal $n, n-$ judgement matrix containing all pair-wise comparisons $\tilde{a}_{ij}$ between elements i and j for all $i, j \in \{1, 2, ..., n\}$

$$\tilde{A} = \begin{bmatrix} (1,1,1) & \tilde{a}_{12} & \dots & \tilde{a}_{1j} \\ \tilde{a}_{12} & (1,1,1) & \dots & \tilde{a}_{12} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{12} & \tilde{a}_{12} & \dots & (1,1,1) \end{bmatrix}$$

where $\tilde{a}_{ij} = \tilde{a}_{ij}^{-1}$ and all $\tilde{a}_{ij}$ are triangular fuzzy numbers, $\tilde{a}_{ij} = \left(l_{ij}, m_{ij}, u_{ij}\right)$. The triangular fuzzy number, $\tilde{a}_{ij}$ consists of l_{ij} and u_{ij} as the lower and the upper limits respectively, and m_{ij} is the point comprising the membership function, $\mu(x) = 1$. The membership functions of the triangular fuzzy number $\mu(x)$ are described as in [29]:

$$\mu(x) = \begin{cases} \dfrac{x-l}{m-l}, & x \in [l,m], \\ \dfrac{x-l}{m-l}, & x \in [l,m], \\ 0, & otherwise \end{cases}$$

where $l_{ij} \le m_{ij} \le u_{ij}$. However, if $l_{ij} = m_{ij} = u_{ij}$ then the fuzzy numbers are the crisp numbers. Originally, the evaluation is performed using Saaty's fundamental scale, which consists of a 9-point scale. The scale ranges from 1 to 9, from equal importance between elements i and j to absolute dominance of i over j and reciprocal values, respectively. However, for this paper, an enhanced scale is used [31]. The scale consists of sets of scale based on linguistic terms and corresponding triangular fuzzy numbers.

Table 2: Linguistic terms and the Corresponding Triangular Fuzzy Numbers [31]

Saaty Scale	Definition	Fuzzy Triangular Scale
1	Equally Important	(1,1,1)
3	Weakly Important	(2,3,4)
5	Fairly Important	(4,5,6)
7	Strongly Important	(6,7,8)
9	Absolutely Important	(9,9,9)
2	The intermediate values between two adjacent scales	(1,2,3)
4		(3,4,5)
6		(5,6,7)
8		(7,8,9)

Via pairwise comparison, the fuzzy evaluation matrix relevant to the goal is constructed in Table 3.

Table 3: The Fuzzy Evaluation Matrix with respect to the goal

	Flexibility	**Performance**	**Reusability**
Flexibility	(1,1,1)	(2,3,4)	(6,7,8)
Performance	(0.25,0.33,0.50)	(1,1,1)	(4,5,6)
Reusability	(0.13,0.14,0.17)	(0.17,0.20,0.25)	(1,1,1)

After the matrix is completed, the consistency ratio of the matrix is measured. The consistency ratio is calculated by using consistency ratio formula. The formula consists of the calculation of the Consistency Index (CI) and Consistency Ratio (CR). The formula is represented as Definition 4.1. CR is calculated to measure the consistency level of judgement relative to large random samples.

Definition 4.1: *Consistency Index (CI) and Consistency Ratio (CR) of fuzzy evaluation matrix.*

$$CI = \frac{\lambda_{\max} - n}{n-1}$$

$$CR = \frac{CI}{RI}$$

where CI is the consistency index, $\lambda_{\max}$ is the largest eigenvalue of matrix, n is the order of comparison matrix, CR is consistency ratio and RI is the random consistency index.

Determination of priority vector is conducted using the eigenvalue approach to determine the desired priority vectors [19]. The weight is derived using a square calculation of the initial pair-wise matrix into a squared weighted matrix. Following that, the values in the matrix undergo a summation and normalisation process before proceeding to obtain the approximation of weight vector. The squared fuzzy evaluation matrix is produced. Next, the values are summed and normalized so as to procure the significant priority vector. Let $X = \{x_1, x_1, \ldots, x_n\}$ be an object set, and $U = \{u_1, u_1, \ldots, u_m\}$ be a goal set. According to the method of extent analysis [29], each object is taken and an extent analysis is performed for each goal respectively. Therefore, m extent analysis values for each object with the following sign are obtained:

$$M_{g_i}^{1}, M_{g_i}^{2}, \ldots, M_{g_i}^{m}, i = 1,2,\ldots,n$$

where all the $M_{g_i}^{j}$ $(j = 1,2,\ldots,m)$ are triangular fuzzy numbers. The value of fuzzy synthetic extent is defined as in Definition 4.2.

Definition 4.2 *Fuzzy Synthetic Extent with respect to the* i^{th} *object*

$$S_i = \sum_{j=1}^{m} M_{g_i}^{j} \otimes \left[\sum_{i=1}^{n} \sum_{j=1}^{m} M_{g_i}^{j} \right]^{-1}$$

From Table 3, by applying Definition 4.2, the following values are obtained:

$$\begin{aligned}
S_{flexibility} &= (9.00, 11.00, 13.00) \otimes (1/21.92, 1/18.67, 1/15.55) \\
&= (9.00, 11.00, 13.00) \otimes (0.046, 0.054, 0.064) \\
&= (0.411, 0.59, 0.836) \\
S_{performance} &= (5.25, 6.33, 7.5) \otimes (1/21.92, 1/18.67, 1/15.55) \\
&= (5.25, 6.33, 7.5) \otimes (0.046, 0.054, 0.064) \\
&= (0.24, 0.34, 0.482) \\
S_{reusability} &= (1.3, 1.34, 1.42) \otimes (1/21.92, 1/18.67, 1/15.55) \\
&= (1.3, 1.34, 1.42) \otimes (0.046, 0.054, 0.064) \\
&= (0.059, 0.07, 0.091)
\end{aligned}$$

The degree of possibility of $M_1 \geq M_2$ is defined as:

$$V(M_1 \geq M_2) = \sup_{x \geq y} \left[\min\left(\mu M_1(x), \mu M_2(y)\right) \right]$$

When a pair (x, y) exists such that $x \geq y$ and $\mu M_1(x) = \mu M_2(y)$, then $V(M_1 \geq M_2) = 1$. Since M_1 and M_2 are convex fuzzy numbers, accordingly:

$$\begin{aligned}
&V(M_1 \geq M_2) = 1 \quad if \quad m_1 \geq m_2, \text{ or} \\
&V(M_1 \geq M_2) = hgt(M_1 \cap M_2) \\
&\qquad\qquad = \mu M_1(d)
\end{aligned}$$

where d is the ordinate of the highest intersection point D between μM_1 and μM_2, as illustrated in Fig. 5.

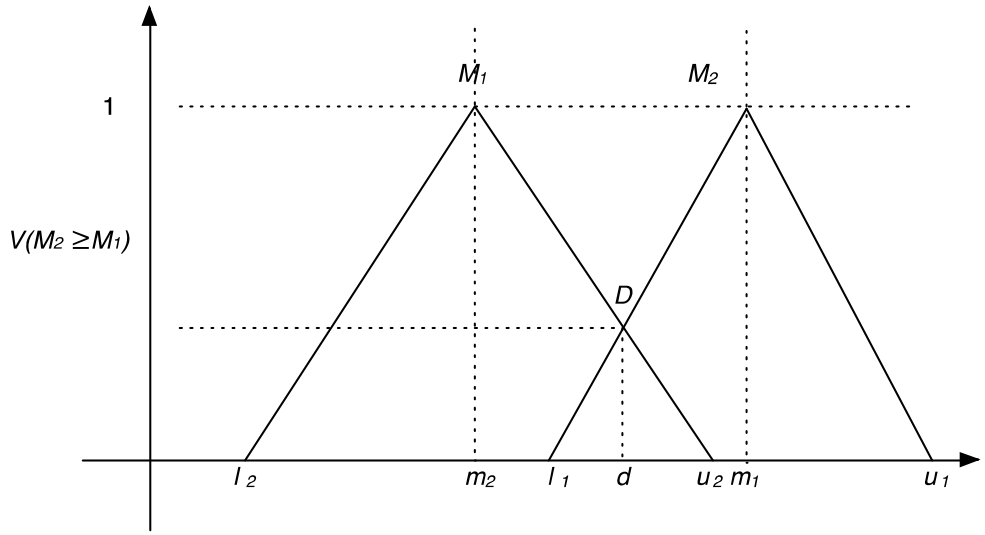

Fig. 5: The intersection between M_1 and M_2

When $M_1 = (l_1, m_1, u_1)$ and $M_2 = (l_2, m_2, u_2)$, the ordinate of D is given by:

$$V(M_1 \geq M_2) = hgt(M_1 \cap M_2)$$
$$= \frac{l_1 - u_2}{(m_2 - u_2) - (m_1 - l_1)}$$

To compare M_1 and M_2 we need both the values of $V(M_1 \geq M_2)$ and $V(M_2 \geq M_1)$. The degree of possibility for a convex fuzzy number to be greater than k convex fuzzy numbers means that $M_i (i = 1,2,\ldots,k)$ can be defined by:

$$V(M \geq M_1, M_2, \ldots, M_k) = V[(M \geq M_1) and (M \geq M_2) and \ldots and (M \geq M_k)]$$

Using these vectors, the values of $V(M_1 \geq M_2)$ and $V(M_2 \geq M_1)$ can be obtained. Hence, the values obtained are:

$$V(S_1 \geq S_2) = 1$$
$$V(S_1 \geq S_3) = 1$$
$$V(S_2 \geq S_1)$$
$$= \frac{0.24 - 0.836}{(0.59 - 0.836) - (0.34 - 0.24)}$$
$$= 1.722$$

$$V(S_2 \geq S_3) = 1$$
$$V(S_3 \geq S_1)$$
$$= \frac{0.059 - 0.836}{(0.59 - 0.836) - (0.07 - 0.059)}$$
$$= 2.995$$
$$V(S_3 \geq S_2)$$
$$= \frac{0.059 - 0.482}{(0.34 - 0.482) - (0.07 - 0.059)}$$
$$= 2.716$$

Finally, let assume that $d'(A_i) = \min V(S_i \geq S_k)$. For $k = 1,2,\ldots,n; k \neq i$. The weight vector is then given by:

$W' = (d'(A_1), d'(A_2), \ldots, d'(A_n))^T$, where $A_i (i = 1,2,\ldots,n)$ are n elements.

Via normalization, the normalized weight vectors are:

$W = (d(A_1), d(A_2), \ldots, d(A_n))^T$, where W is a non-fuzzy number. Therefore, these values are obtained:

$$d'(Flexibility) = V(S_1 \geq S_2, S_3)$$
$$= \min(1,1)$$
$$= 1$$
$$d'(Performance) = V(S_2 \geq S_1, S_3)$$
$$= \min(1.722,1)$$
$$= 1$$
$$d'(\text{Re}usability) = V(S_3 \geq S_1, S_2)$$
$$= \min(2.995, 2.716)$$
$$= 2.716$$

Therefore, $W' = (1,1,2.716)^T$ and via normalization, the weight vectors obtained with respect to the criteria *Flexibility, Performance* and *Reusability* are $W = (0.212, 0.212, 0.576)^T$.

The evaluation process then compares the sub-criteria with respect to main criteria. The other tables will not be given in the paper, as the calculation is similar. Finally, after adding the weights for goal alternatives multiplied by the weight of the corresponding criteria, a final score is obtained for each software

architecture style. Table 4 shows the final scores for the software architecture style. A layered-style software architecture style is selected for the design.

Table 4: The Final Scores

	Centralized	Client-Server	Layered
Final Scores	0.13	0.40	0.47

5 Design Phase

Based on the PF Reference Architecture, which has been built using the enhanced SPL method, an initial PF software architecture is developed using the selected software architecture style. Component-Based Development (CBD) has been chosen as one of the layered software architecture styles catering to the initial design of PF system. The component is then modelled using the integrated component model proposed in [14]. The component model is a part of the Code Generation Implementation Steps© and implemented using a Component Oriented Programming (COP) framework [30]. Fig. 5 shows an example of the PF system modelled using layered architecture style, taking advantage of the component model.

In COP Framework, the component model specifications component compositions are defined in an integrated component model form. This integrated component model is a part of the proposed Code Generation Implementation Steps©. The Code Generation Implementation Steps process is proposed as a guideline for generating codes and implements through a commercial software development tool. The code generation implementation steps are made up of several steps, represented as follows.

The components are modeled using the proposed integrated component model. Later, the component models are mapped into the software modeling tool. After components specification, these components are realized using two artifacts, namely: class diagram and state diagram. Both of these models are also mapped into the software modeling tool. Subsequently, the component models, diagrams, class diagrams and state diagrams respectively are verified as being either correct or incorrect. This checking process can ensure that all the diagrams are mapped accurately into the modeling tool. Eventually, the components are composed and integrated in the Composite Component diagram. The components composition and integration process is significant by which to develop a whole system. In addition, this process checks the diagram correctness.

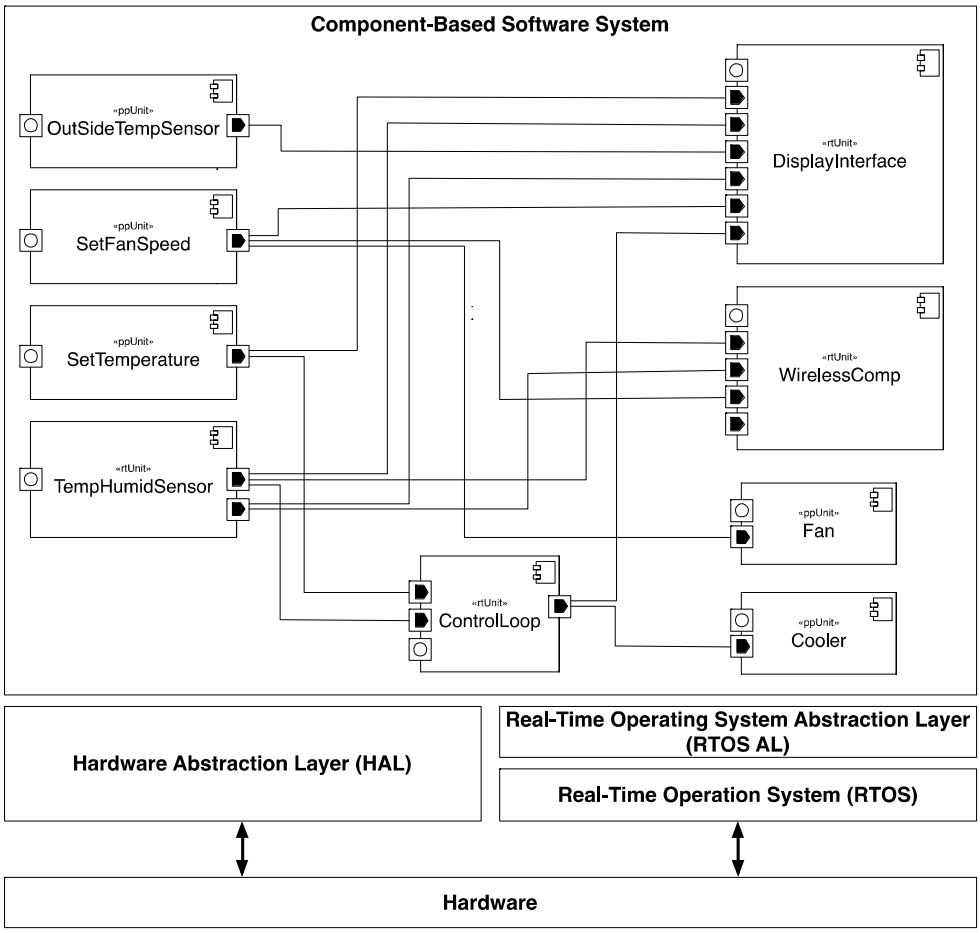

Fig. 5: Initial Design of PF System using Layered Architecture Style

Referring to PF Reference Architecture as shown in Fig. 3, a component composition diagram is then constructed. In the diagram for the PF reference architecture, such examples of hardware components are identified, specifically: `OutSideTempSensor`, `TempHumidSensor`, `Fan`, `Cooler`, `DisplayInterface` and `WirelessComp`, while the software components are `SetFanSpeed`, `SetTemperature` and `ControlLoop`. These hardware components are identified from the PF reference architecture, considering hardware and software relations respectively.

The HAL provides some interface functions for controlling actuators and reading sensors on the mobile robot. Thus, a user without any hardware knowledge can program the mobile robot easily. Fig. 5 shows how HAL and RTOS connect to each other so as to provide interface to the related hardware devices.

HAL provides decoupling between the application software and the underlying hardware. RTOS provides an abstraction layer that hides from application software the hardware details of the processor or set of processors, upon which the application software will run. In the development of real-time embedded systems,

the use of RTOS will increase the software productivity and improve the performance of the real-time system.

The RTOS Abstraction Layer (RTOS AL) provides a thin layer of interface between the components and the actual RTOS. Much of the software code described in HAL is written so that the software engineer does not need to know in detail what hardware devices are used or how to connect and interact with that hardware. Instead, abstracted functionality is provided in HAL in order to promote software reuse. Besides, this also can help to simplify the software coding process.

6 Conclusion

The integration of an enhanced SPL method and a multi-criteria decision-making method shows some promises to counter the complexity in PF system development by aiding the selection of a suitable software architecture style through the application of enhanced FArM methods for analysis. Further, it is assisted by fuzzy AHP for the selection process. From the selection process, a suitable software architecture style is chosen based on the criteria provided.

From the results obtained from the analysis phase to the design phase, it can be seen that the current effort of choosing and selecting a suitable software architectural style has been reduced. With the introduction of the fuzzy AHP, a suitable software architecture design is selected for aiding the design process. Using multi-criteria analytic hierarchy process and fuzzy logic mechanism offered by fuzzy AHP methods for selection, the PF system could be developed correctly using suitable software architecture style to meet the PF system requirements.

ACKNOWLEDGEMENTS

Special thanks are extended to the Universiti Teknologi Malaysia for providing facilities, support, and guidance; also to our Embedded & Real-Time Software Engineering Laboratory (EReTSEL) and Software Engineering Research Group (SERG), K-Economy Research Alliance (RAKE), Universiti Teknologi Malaysia members for their continuous support.

References

[1] W. B. Frakes and K. Kang. 2005. Software Reuse Research: Status and Future, *IEEE Transactions on Software Engineering*, Vol. 31, No. 7, 529-536.

[2] K. Pohl, G. Böckle and F. V. Linden. 2005. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer-Verlag New York Inc.

[3] H. Gomaa. 2004. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison-Wesley Longman Publishing Inc.

[4] H. Mili, A. Mili, S. Yacoub and E. Addy. 2001. *Reuse-Based Software Engineering: Techniques, Organization, and Controls*. Wiley-Interscience New York.

[5] K. C. Kang, S.G. Cohen, J. A. Hess, W. E. Novak and A. S. Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Carnegie Mellon University, Software Engineering Institute. CMU/SEI-90-TR-21.

[6] K. Lee, K. C. Kang, W. Chae and B. W. Choi. 2000. Feature-Based Approach to Object-Oriented Engineering of Applications for Reuse. *Software-Practice and Experience,* Vol. 30, No. 9, 1025-1046

[7] K.C. Kang, S. Kim, J. Lee, K. Kim, E. Shin and M. Huh. 1998. FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. *Annals Of Software Engineering*. Vol. 5, No. 1. 143-168.

[8] P. Sochos, M. Riebisch and I. Philippow. 2006. The Feature-Architecture Mapping (FArM) Method for Feature-Oriented Development of Software Product Lines, *13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based System (ECBS 2006)*, 318.

[9] M. Matinlassi. 2004. Comparison of Software Product Line Architecture Design Method: COPA, FAST, FORM, KobrA, and QADA, *26th International Conference on Software Engineering (ICSE 2004),* 127-136, IEEE.

[10] P. Sochos. 2007. The Feature-Architecture Mapping Method for Feature-Oriented Development of Software Product Lines. Doctor of Philosophy.

[11] K. Lee and K.C. Kang. 2004. Feature Dependency Analysis for Product Line Component Design, *Software Reuse: Methods, Techniques and Tools*, Vol. 3107, 69-85

[12] K. C. Kang, J. Lee and P. Donohoe. 2002. Feature-Oriented Product Line Engineering, *IEEE Software*. Vol. 19, No. 4, 58-65.

[13] N.M. Hamdan. 2012. The Integrated Method for a Systematic Approach in Developing Precision Farming Software Product Line Architecture, *Master of Science (Computer Science)*, Universiti Teknologi Malaysia.

[14] M. Z. M. Zaki. 2012. Integrated Component-Based Model and Code Generation Implementation Steps for Embedded Real-Time System Development, *Master of Science (Computer Science)*, Universiti Teknologi Malaysia.

[15] S. Vijayalakshmi, G. Zayaraz and V. Vijayalakshmi. 2010. Multicriteria Decision Analysis Method for Evaluation of Software Architectures, *International Journal of Computer Applications*, Vol. 1, No. 25, 22-27.

[16] T. L. Saaty. 1980. *The Analytic Hierarchy Process,* McGraw-Hill International, New York.

[17] C. Kahraman, D. Ruan and I. Doğan. 2003. Fuzzy group decision-making for facility location selection, *Information Sciences*, Vol. 157, 135-153.

[18] C. Kahraman, U. Cebeci and Z. Ulukan. 2003. Multi-criteria supplier selection using fuzzy AHP, *Logistics Information Management*, Vol. 16, No. 6, 382-394.

[19] T. L. Saaty. 2008. Decision making with the analytic hierarchy process, *International Journal of Services Sciences*, Vol. 1, No. 1, 83-98.

[20] S. D. Kim, H. G. Min, J. S. Her and S. H. Chang. 2005. DREAM: A practical product line engineering using model driven architecture. *The Third International Conference on Information Technology and Application (ICITA 2005),* Vol. 1, 70-75, IEEE.

[21] S. Moven, J. Habibi, H. Asmadi and A. Kamandi. 2008. A Fuzzy Model for Solving Architecture Styles Selection Multi-Criteria Problem*, Second UKSIM European Symposium Computer Modelling and Simulation (EMS 2008),* 288-293, IEEE.

[22] L. Tan, Y. Lin, and H. Ye. 2012. Modeling Quality Attributes in Software Product Line Architecture, *2012 Spring Congress on Engineering and Technology (S-CET)*, 1–5, IEEE.

[23] J. D. Meier. 2009. A Language for Software Architecture. *The Architecture Journal TechEd Special Edition.* Microsoft.

[24] L. S. Iliadis, R. Nikkilä, I. Seilonen and K. Koskinen. 2010. Software architecture for farm management information systems in precision agriculture. *Computers and Electronics in Agriculture*, Vol. 70, No. 2, 328–336.

[25] Y. Wang, Y. Wang, X. Qi and L. Xu. 2009. OPAIMS: Open Architecture Precision Argiculture Information Monitoring System, *Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES 2009)*, 233-240, ACM Press.

[26] X. Li, Y. Deng and L. Ding. 2008. Study on precision agriculture monitoring framework based on WSN. *2nd International Conference on Anti-counterfeiting, Security and Identification (ASID 2008)*. 182–185, IEEE.

[27] R. Jaichandran. 2011. Prototype System for Monitoring and Computing Greenhouse gases. *World of Computer Science and Information Technology Journal (WCSIT)*, Vol. 1, No. 5, 177–183.

[28] M. Galster, A. Eberlein and M. Moussavi. 2010. Systematic selection of software architecture styles. *IET Software*, Vol. 4, No. 5, 349-360.

[29] D. Y. Chang. 1996. Applications of the extent analysis method on fuzzy AHP, *European Journal of Operational Research,* Vol. 95, No. 3, 649-655.

[30] D. N. A. Jawawi, S. Deris, and R. Mamat. 2008. Early-Life Cycle Reuse Approach for Component-Based Software of Autonomous Mobile Robot System. *The 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2008),* 263-268, IEEE.

[31] M. B. Ayhan. 2013. A Fuzzy AHP Approach for Supplier Selection Problem: A Case Study in a GearMotor Company. *International Journal of Managing Value and Supply Chains (IJMVSC),* Vol. 4, No. 3, 11-23.